CDAT Gotchas

Common pitfalls

Single−precision arithmetic

Using single−precision floating point arrays has the advantage of saving considerable space for very large arrays, but there are a few pitfalls to be aware of. Consider the following:

```
>>> import Numeric
>>> x = Numeric.ones(20000000., Numeric.Float32)
>>> Numeric.average(x)
0.83886079999999996
>>>
```

This is a huge bug, right???

Surprising, yes, but a bug, no. In fact, this is the result that should be expected for 32−bit floats, which have a 23−bit mantissa and 8−bit exponent. The largest integer that can be represented exactly as a 32−bit float is 2**24 = 16777216.0 . The Numeric **average** function accumulates the sum in a single−precision float, which eventually reaches the value 16777216.0. At that point, the next addition has the result:

16777216.0 + 1.0 = 16777216.0

hence the non−intuitive result.

The moral of the story is: for very large arrays do the calculations in double precision, especially where sums and averages are taken.

Setting values of a CDMS unlimited file axis, or a file variable with an unlimited axis

It is often necessary to replace the values of an unlimited axis of a file. Ordinarily this is the time axis of a spatiotemporal dataset. An *unlimited* axis has the property that data can be written beyond the current length of the axis, in order to extend a datafile. To replace the values of a time axis it is natural to try the following:

```
>>> import cdms
>>> f = cdms.open('sample.nc')
>>> t = f['time']
>>> t.isUnlimited()
True
>>> x = t[:] + 10000.
>>> x
[ 83063.5, 83093.5, 83123.5, 83154. ,]
>>> t[:] = x[:]
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/export/data/cdat/cdat-4.0b7/lib/python2.4/site-packages/cdms/axis.py", line 1756, in __s
    return apply(self._obj_.setslice,(low,high,MA.filled(value)))
ValueError: shapes are not aligned
>>>
```

What's going on?

The expression **t[:]** is shorthand for **t[0:lastindex]** where lastindex−1 is the largest valid index for **t**. But if **t** is unlimited, there IS no largest index. In fact, lastindex resolves to a system−dependent value, typically 2^31 = 2147483648. So internally the statement

```
t[:] = x[:]
```

becomes

```
t[0:2147483648] = x[0:4]
```

Since the shapes do not match, an exception is raised. The correct way is to use the fact that len(t) returns the length of t as currently written to the file:

```
>>> t[:len(t)] = x[:]
>>>
```

Similar considerations apply to setting the values of a file variable having an unlimited axis in its domain. For example:

```
>>> import cdms, MV
>>> f = cdms.open('hfogo_O1.nc','r+')
>>> hfogo = f['hfogo']
>>> hfogo.getAxisIds()
['time', 'region', 'lat']
>>> time = f['time']
>>> time.isUnlimited()
True
>>> hfogo.shape
(1812, 4, 61)
>>> len(hfogo)  # Length of the first dimension
1812
#
# First try. Result is a double, which must be recast
#
>>> hfogo[:] = 1000*hfogo[:]
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/export/data/cdat/cdat-4.0b7/lib/python2.4/site-packages/cdms/fvariable.py", line 100, in
    apply(self._obj_.setslice,(low,high,MA.filled(value)))
TypeError: Array can not be safely cast to required type
#
# Second try, casting the result to single precision
#
>>> hfogo[:] = (1000*hfogo[:]).astype(MV.Float32)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/export/data/cdat/cdat-4.0b7/lib/python2.4/site-packages/cdms/fvariable.py", line 100, in
    apply(self._obj_.setslice,(low,high,MA.filled(value)))
ValueError: shapes are not aligned
#
# The correct way
#
>>> hfogo[:len(hfogo)] = (1000*hfogo[:]).astype(MV.Float32)
>>>
```